**AMENDMENTS TO THE CLAIMS:**

Please cancel without prejudice claims 3, 6, 10, 15, 18, 22, 27, 30 and 34 and amend claims 1, 4, 5, 7-9, 11-13, 19-21, 23-25, 28, 29, 31-33, 35 and 36 as follows.

This listing of claims will replace all prior versions, and listings, of claims in the application:

1. (currently amended) A computer program product for controlling a computer to ~~detecting~~detect an executable computer program containing a computer virus, said computer program product comprising:

analysis logic ~~operable to analyse~~for analyzing program instructions forming said executable computer program to identify suspect program instructions being at least one ~~or more~~ of:

(i) a program instruction generating a result value not used by another portion of said executable computer program; and

(ii) a program instruction dependent upon an uninitialised variable; and

detecting logic ~~operable to detect~~for detecting said executable computer program as containing a computer virus if a number of suspect program instructions identified for said executable computer program exceeds a threshold level, wherein said analysis logic includes a dependence table indicating dependency between state variables within said computer and loaded variable values, and for each program instruction said analysis logic makes a determination as to which state variables are read and written by that program instruction and for each loaded variable value within said dependence table if any state variable read by that program instruction is marked as dependent upon said loaded variable value, then all state

variables written by that program instruction are marked as dependent upon said loaded variable value with previous dependencies being cleared, and said analysis logic parses said executable computer program for suspect program instructions by following execution flow and upon occurrence of a branch first following a first branch path having saved pending analysis results and subsequently returning to follow a second branch path having restored said pending analysis results.

2. (original) A computer program product as claimed in claim 1, wherein said computer virus is a polymorphic computer virus.

3. (cancelled).

4. (currently amended) A computer program product as claimed in claim 1, wherein for each program instruction said analysis logic ~~is operable to make~~makes a determination as to which state variables are read by that program instruction.

5. (currently amended) A computer program product as claimed in claim 1, wherein for each program instruction said analysis logic ~~is operable to make~~makes a determination as to which state variables are written by that program instruction.

6. (cancelled).

7. (currently amended) A computer program product as claimed in claim ~~3~~1, wherein said state variables include at least one ~~or more~~ of:

(i) register values;

(ii) processing result flag values; and

(iii) a flag indicative of a write to a non-register storage location.

8. (currently amended) A computer program product as claimed in claim 1, wherein said analysis logic ~~is operable to maintain~~includes an initialisation table indicating which state variables have been initialised.

9. (currently amended) A computer program product as claimed in claim 8, wherein a state variable is marked as initialised upon occurrence of ~~any~~ one of:

(i) a write to said state variable of a determined initialised value; and

(ii) use of said state variable as a memory address value by a program instruction.

10. (cancelled).

11. (currently amended) A computer program product as claimed in claim ~~10~~1, wherein a branch path stops being followed when ~~any~~ one of the following occurs:

(i) there are no further suitable program instruction for execution within that branch path; and

(ii) said branch path rejoins a previously parsed execution path.

12. (currently amended) A computer program product as claimed in claim 1, wherein if said threshold level is exceeded, then further virus detection mechanisms are triggered to confirm the presence of a computer virus.

13. (currently amended) A method of detecting an executable computer program containing a computer virus, said method comprising the steps of:

analysing program instructions forming said executable computer program to identify suspect program instructions being at least one ~~or more~~ of:

(i) a program instruction generating a result value not used by another portion of said executable computer program; and

(ii) a program instruction dependent upon an uninitialised variable; ~~and~~

detecting said executable computer program as containing a computer virus if a number of suspect program instructions identified for said executable computer program exceeds a threshold level;

maintaining a dependence table indicating dependency between state variables within said computer and loaded variable values, wherein for each program instruction a determination is made as to which state variables are read and written by that program instruction and, for each loaded variable value within said dependence table, if any state variable read by that program instruction is marked as dependent upon said loaded variable value, then all state variables written by that program instruction are marked as dependent upon said loaded variable value with previous dependencies being cleared; and

parsing said executable computer program for suspect program instructions by following execution flow and upon occurrence of a branch first following a first branch path having saved

pending analysis results and subsequently returning to follow a second branch path having restored said pending analysis results.

14. (original) A method as claimed in claim 13, wherein said computer virus is a polymorphic computer virus.

15. (cancelled).

16. (original) A method as claimed in claim 13, wherein for each program instruction a determination is made as to which state variables are read by that program instruction.

17. (original) A method as claimed in claim 13, wherein for each program instruction a determination is made as to which state variables are written by that program instruction.

18. (cancelled).

19. (currently amended) A method as claimed in claim ~~15~~13, wherein said state variables include at least one ~~or more~~ of:

(i) register values;

(ii) processing result flag values; and

(iii) a flag indicative of a write to a non-register storage location.

20. (currently amended) A method as claimed in claim 13, including the step of maintaining ~~wherein~~ an initialisation table ~~is maintained~~ indicating which state variables have been initialised.

21. (currently amended) A method as claimed in claim 20, wherein a state variable is marked as initialised upon occurrence of ~~any~~ one of:

(i) a write to said state variable of a determined initialised value; and

(ii) use of said state variable as a memory address value by a program instruction.

22. (cancelled).

23. (currently amended) A method as claimed in claim ~~22~~13, wherein a branch path stops being followed when ~~any~~ one of the following occurs:

(i) there are no further suitable program instruction for execution within that branch path; and

(ii) said branch path rejoins a previously parsed execution path.

24. (currently amended) A method as claimed in claim 13, wherein if said threshold level is exceeded, then further virus detection mechanisms are triggered to confirm the presence of a computer virus.

25. (currently amended) Apparatus for detecting an executable computer program containing a computer virus, said apparatus comprising:

an analyser ~~operable to analyse~~for analysing program instructions forming said executable computer program to identify suspect program instructions being at least one ~~or more~~ of:

(i) a program instruction generating a result value not used by another portion of said executable computer program; and

(ii) a program instruction dependent upon an uninitialised variable; and

a detector operable to detect said executable computer program as containing a computer virus if a number of suspect program instructions identified for said executable computer program exceeds a threshold level, wherein said analyser includes a dependence table indicating dependency between state variables within said computer and loaded variable values, wherein for each program instruction said analyser makes a determination as to which state variables are read and written by that program instruction and for each loaded variable value within said dependence table if any state variable read by that program instruction is marked as dependent upon said loaded variable value, then all state variables written by that program instruction are marked as dependent upon said loaded variable value with previous dependencies being cleared, and said analyser parses said executable computer program for suspect program instructions by following execution flow and upon occurrence of a branch first following a first branch path having saved pending analysis results and subsequently returning to follow a second branch path having restored said pending analysis results.

26. (original) Apparatus as claimed in claim 25, wherein said computer virus is a polymorphic computer virus.

27. (cancelled).

28. (currently amended) Apparatus as claimed in claim 25, wherein for each program instruction said analyser ~~is operable to make~~makes a determination as to which state variables are read by that program instruction.

29. (currently amended) Apparatus as claimed in claim 25, wherein for each program instruction said analyser ~~is operable to make~~makes a determination as to which state variables are written by that program instruction.

30. (cancelled).

31. (currently amended) Apparatus as claimed in claim ~~27~~25, wherein said state variables include at least one ~~or more~~ of:

(i) register values;

(ii) processing result flag values; and

(iii) a flag indicative of a write to a non-register storage location.

32. (currently amended) Apparatus as claimed in claim 25, wherein said analyser ~~is operable to maintain~~includes an initialisation table indicating which state variables have been initialised.

33. (currently amended) Apparatus as claimed in claim 32, wherein a state variable is marked as initialised upon occurrence of ~~any~~ one of:

(i) a write to said state variable of a determined initialised value; and

(ii) use of said state variable as a memory address value by a program instruction.

34. (cancelled).

35. (currently amended) Apparatus as claimed in claim ~~34~~25, wherein a branch path stops being followed ~~when any~~upon the occurrence of one of:

(i) there are no further suitable program instruction for execution within that branch path; and

(ii) said branch path rejoins a previously parsed execution path.

36. (currently amended) Apparatus as claimed in claim 25, wherein if said threshold level is exceeded, then further virus detection mechanisms are triggered to confirm the presence of a computer virus.